

Introdução a SQL

Prof. Anderson Cavalcanti

UFRN-CT-DCA

Características da **Structured** **Query Language - SQL**

Características da SQL

- SQL é uma linguagem de pesquisa declarativa para banco de dados relacional. Muitas das características originais do SQL foram inspiradas na álgebra relacional;
- Foi desenvolvida originalmente no início dos anos 70 nos laboratórios da IBM em *San Jose* e *tinha por objetivo demonstrar* a viabilidade da implementação do modelo relacional proposto por E. F. Codd;
- O nome original da linguagem era SEQUEL, acrônimo para "Structured English Query Language".

Características da SQL

- A linguagem SQL é um grande padrão de banco de dados. Isto decorre da sua simplicidade e facilidade de uso.
- Ela é uma linguagem declarativa em oposição a outras linguagens procedurais. Isto reduz o ciclo de aprendizado daqueles que se iniciam na linguagem.

Características da SQL

- Divisão da linguagem SQL:
 - **Linguagem de Definição de Dados (DDL):** A DDL da SQL fornece comandos para definir esquemas de relação, excluir relações e modificar esquemas;
 - **Linguagem de Manipulação de Dados (DML):** A DML inclui uma linguagem de consulta. Também possui comandos para inserir, excluir e modificar dados no BD;

Características da SQL

- Divisão da linguagem SQL:
 - **Linguagem de Controle de Dados (DCL):** Controla os aspectos de autorização de dados e licenças de usuários para controlar quem tem acesso para ver ou manipular dados dentro do banco de dados.
 - **Linguagem Transação de Dados (DTL):** Controla as transações do Banco de Dados.

Características da SQL

- **Instruções da DML:**

- **SELECT:** Instrução que permite ao usuário especificar uma consulta como uma descrição do resultado desejado.
- **INSERT:** Instrução que é usada para inserir um registro numa tabela existente.
- **UPDATE:** Instrução que altera os valores de dados em um registro da tabela especificada.
- **DELETE:** Instrução que permite remover registros existentes de uma tabela.

Características da SQL

- **Instruções da DDL:**
 - **CREATE:** Instrução que cria um objeto (uma tabela, por exemplo) dentro da base de dados.
 - **DROP:** Instrução que apaga um objeto do banco de dados.
- Alguns sistemas de banco de dados usam o comando **ALTER**, que permite ao usuário alterar um objeto, por exemplo, adicionando uma coluna a uma tabela existente.

Características da SQL

- **Instruções da DCL:**

- **GRANT:** Instrução que autoriza ao usuário executar ou configura operações.
- **REVOKE:** Instrução que remove ou restringe a capacidade de um usuário de executar operações.

Características da SQL

- Instruções da DTL:
 - **BEGIN WORK** (ou **START TRANSACTION**, dependendo do dialeto SQL) pode ser usado para marcar o começo de uma transação de banco de dados que pode ser completada ou não.
 - **COMMIT** envia todos os dados das mudanças permanentemente.
 - **ROLLBACK** faz com que as mudanças nos dados existentes desde que o último COMMIT ou ROLLBACK sejam descartadas.

SQL - DDL

Tipos de Domínio Básicos

- **char(n)**: uma string de caracteres de tamanho fixo, com tamanho n;
- **varchar(n)**: uma string de caracteres de tamanho variável, com tamanho máximo n;
- **int ou integer**: um inteiro (depende da máquina);
- **smallint**: um inteiro pequeno;

Tipos de Domínio Básicos

- **numeric(p,q):** um número de ponto fixo com precisão especificada pelo usuário. São p dígitos dos quais q deles estão depois da vírgula.
 - **Exemplo:** numeric(3,1) permite a representação do número 22,5 e não permite a representação do número 0,31 nem do número 214,2.
- **real, double precision:** números de ponto flutuante e ponto flutuante de precisão dupla, com precisão dependente da máquina.
- **float(n):** um número de ponto flutuante, com precisão de pelo menos n dígitos.

Criando Tabelas e Chaves

- Instrução **CREATE**

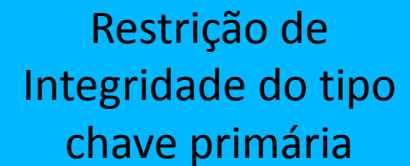
```
create table nome_tabela(  
    atributo_1 tipo_domínio_1,  
    atributo_2 tipo_domínio_2, ...,  
    atributo_n tipo_domínio_n,  
    restrição_integridade_1,  
    restrição_integridade_2, ...,  
    restrição_integridade_n  
)
```

Criando Tabelas e Chaves

- Exemplo:

Cliente (id_cliente, nome_cliente, endereço)

```
create table cliente (  
    id_cliente integer,  
    nome_cliente varchar(60),  
    endereco varchar(60),  
    primary key(id_cliente)  
)
```



Restrição de
Integridade do tipo
chave primária

Criando Tabelas e Chaves

- Exemplo:

Empregado(id_empregado, id_departamento, nome_empregado)
id_departamento referencia Departamento

Departamento(id_departamento, nome_departamento)

Criando Tabelas e Chaves

```
create table departamento (  
    id_departamento integer,  
    nome_departamento varchar(60),  
    primary key(id_departamento)  
)  
  
create table empregado (  
    id_empregado integer, id_depto integer,  
    nome_empregado varchar(60),  
    primary key(id_empregado),  
    foreign key(id_depto) references  
    departamento(id_departamento)  
)
```

Criando Tabelas e Chaves

- Exemplo:

Locação(id_locação, data);

DVD(id_dvd,título,gênero)

Item_Locação(id_locação, id_dvd) id_locação referencia Locação, id_dvd referencia DVD

Criando Tabelas e Chaves

```
create table dvd (  
    id_dvd integer,  
    titulo varchar(100),  
    genero varchar(60),  
    primary key(id_dvd)  
)
```

```
create table locacao (  
    id_locacao integer,  
    data varchar(10),  
    primary key(id_locacao)  
)
```

```
create table item_locacao (  
    id_dvd integer,  
    id_locacao integer,  
    primary key(id_locacao, id_dvd),  
    foreign key(id_dvd) references dvd(id_dvd),  
    foreign key(id_locacao) references locacao(id_locacao)  
)
```

Criando Tabelas e Chaves

- Restrições já apresentadas:
 - Restrição de chave primária (primary key);
 - Restrição de integridade referencial (foreign key);
- Outras Restrições:
 - Integridade de Vazio - not null;
 - Integridade de Chave Alternativa - unique;
 - Restrição Semântica - check(<predicado>).

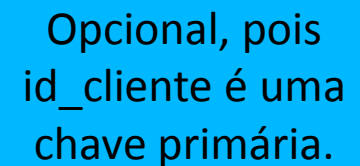
Integridade de Vazio - NOT NULL

- Foi estudado que certos atributos de entidade podem ser nulos (ex.: atributo numero_apartamento para a entidade cliente);
- Ao usarmos a restrição **not null**, estamos afirmando que o determinado atributo não poderá receber valor nulo;
- O SGBD gera um erro se esse tipo de restrição não for obedecida.

Integridade de Vazio - NOT NULL

- Exemplo da Restrição **NOT NULL**

```
create table cliente (  
    id_cliente integer not null,  
    nome_cliente varchar(60) not null,  
    endereco varchar(60),  
    primary key(id_cliente)  
)
```



Opcional, pois
id_cliente é uma
chave primária.

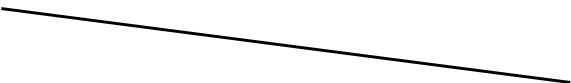
Integridade de Chave Alternativa - UNIQUE

- A especificação **unique (A1,...,An)** diz que os atributos **A1,...,An** formam uma chave alternativa, ou seja, nenhum par de entidades/relacionamentos pode ser igual em todos os atributos;
- Entretanto, os atributos de chave alternativa podem ser nulos, a menos que tenham sido declarados como **not null**;
- Devemos lembrar que o valor nulo não se iguala a qualquer outro valor.

Integridade de Chave Alternativa - UNIQUE

- Exemplo da Restrição **UNIQUE**

```
create table cliente (  
    id_cliente integer,  
    cpf varchar(14),  
    nome_cliente varchar(60) not null,  
    endereco varchar(60),  
    primary key(id_cliente),  
    unique(cpf)  
)
```



Chave alternativa.

Restrição Semântica - CHECK

- A cláusula **check** pode ser aplicada a declarações de tabelas, bem como a declarações de **domínios**;
- Exemplo em declarações de tabelas:

```
create table aluno (  
    id_aluno integer,  
    nome varchar(60) not null,  
    nivel_grau varchar(15),  
    primary key(id_aluno),  
    check (nivel_grau in  
        ('Bacharelado', 'Mestrado', 'Doutorado'))  
)
```

Restrição Semântica - CHECK

- Exemplo em declarações de domínios:

```
create domain salario numeric(15,2)  
    check(value >= 465)
```

Alterando Tabelas Existentes

- Adicionando nova coluna:

```
alter table nome_tabela add nova_coluna  
dominio_nova_coluna
```

- Exemplo:

```
alter table cliente add email  
varchar(255)
```

Alterando Tabelas Existentes

- Excluindo atributo existente:

```
alter table nome_tabela drop  
    atributo_existente
```

- Exemplo:

```
alter table cliente drop email
```

Excluindo Tabelas Existentes

- Excluindo Tabela Existente

```
drop table nome_tabela
```

- Exemplo:

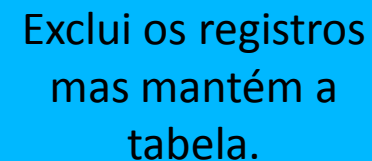
```
drop table cliente
```

- Excluindo TODOS os registros da Tabela

```
drop from nome_tabela
```

- Exemplo:

```
drop from cliente
```



Exclui os registros
mas mantém a
tabela.

SQL - DML

Inserindo Dados Numa Tabela Existente

- Instrução **INSERT**

```
insert into nome_tabela (  
    atributo_1,  
    atributo_2, ...,  
    atributo_n  
) values (  
    valor_atributo_1,  
    valor_atributo_2, ...,  
    valor_atributo_n  
)
```

Inserindo Dados Numa Tabela Existente

- Exemplo:

```
insert into cliente (  
    id_cliente,  
    nome,  
    endereco  
) values (  
    1,  
    "Fulano dos Anzóis",  
    "Rua do Sol, 32"  
)
```


Consultando Tabelas

- Instrução **SELECT**

```
select C1, C2, ..., Cn from T1, T2, ..., Tm  
where P
```

- Em que:

- C_i → Coluna i;
- T_j → Tabela j;
- P → Predicado

Consultando Tabelas

- Exemplo:

Tabela: aluno

<u>id_aluno</u>	nome_aluno
1	TIRIRICA
2	ABILOALDO TIMÓTEO

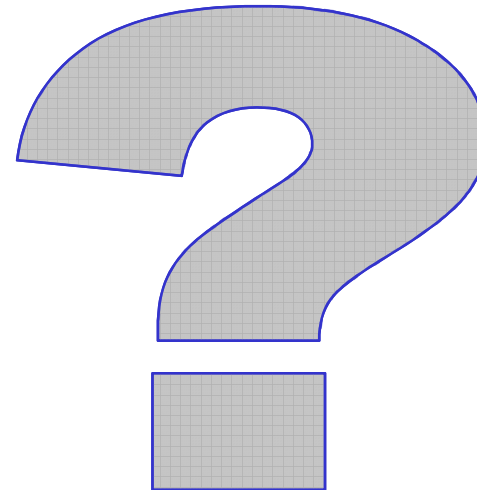
Tabela: disciplina

<u>id_disciplina</u>	nome_disciplina
1	ALGORITMO

Tabela: curso

<u>id_curso</u>	nome_curso
1	SI

```
select nome_aluno,  
       nome_disciplina,  
       nome_curso from  
aluno, disciplina,  
curso
```



Consultando Tabelas

- A consulta acima equivale a, primeiro, efetuar a operação aluno x disciplina x curso:

<u>id_aluno</u>	nome_aluno	<u>id_disciplina</u>	nome_disciplina	<u>id_curso</u>	nome_curso
1	TIRIRICA	1	ALGORITMO	1	SI
2	ABILOALDO TIMÓTEO	1	ALGORITMO	1	SI

- Da tabela acima, escolhe apenas as colunas especificadas no select:

nome_aluno	nome_disciplina	nome_curso
TIRIRICA	ALGORITMO	SI
ABILOALDO TIMÓTEO	ALGORITMO	SI

Para especificar todas as colunas, pode-se utilizar o caracter *

Cláusula Where

- Exemplo:

```
select nome_aluno, nome_disciplina, nome_curso from  
aluno, disciplina, curso where  
nome_aluno="TIRIRICA"
```

nome_aluno	nome_disciplina	nome_curso
TIRIRICA	ALGORITMO	SI

Cláusula Where

- A cláusula where pode conter diversos conectivos lógicos como **and**, **not** e **or**.
- A cláusula ainda suporta diversos operadores de comparação como <, <=, >, >=, = e <>.

Cláusula Where

- Considere a tabela cliente:

<u>id_cliente</u>	nome_cliente	credito
1	José de Melo Bico	1000
2	Antônio Barbudinho	800
3	Waldick Soriano	1200
4	Tiririca da Silva	350

```
select * from cliente where credito >= 500 and  
credito <= 1000
```

<u>id_cliente</u>	nome_cliente	credito
1	José de Melo Bico	1000
2	Antônio Barbudinho	800

Operações com Strings

- As operações em strings mais usadas são as checagens para verificação de coincidências de pares, utilizando o operador LIKE. Esses pares são identificados por meio do uso de dois caracteres especiais:
- Porcentagem (%): compara qualquer string;
- Sublinhado (_): compara qualquer caractere.

Operações com Strings

- Exemplos:
- "_ _ _ _%" corresponde a qualquer string com pelo menos quatro caracteres.
- "Uni%" corresponde a qualquer string que comece com "Uni", como, "universo", "universal", "universidade".
- Utilizando not LIKE pode-se pesquisar diferenças, ao invés de coincidências.
- Obs.: Essas comparações são case sensitive.

Distinct

- SQL permite duplicidades nas tuplas de resposta. Quando desejamos forçar a eliminação de duplicidade, podemos inserir a palavra chave DISTINCT depois de SELECT.

<u>id_cliente</u>	nome_cliente	bairro
1	José de Melo Bico	Potengi
2	Antônio Barbudinho	Nova Natal
3	Waldick Soriano	Nova Natal
4	Tiririca da Silva	Lagoa Nova

```
select distinct(bairro) from cliente
```

bairro
Potengi
Nova Natal
Lagoa Nova

Ordenação na Exibição de Registros

<u>id_cliente</u>	nome_cliente	credito
3	Waldick Soriano	1200
4	Tiririca da Silva	350
1	José de Melo Bico	1000
2	Antônio Barbudinho	800

```
select nome_cliente, credito from cliente order  
by nome_cliente desc
```

Funções Agregadas

- As funções agregadas são aquelas que tomam uma coleção de valores como entrada e retornam um único valor;
- A SQL oferece cinco funções básicas embutidas:
 - Média (average): avg;
 - Mínimo: min;
 - Máximo: max;
 - Soma: sum;
 - Conta: count.

Funções Agregadas

- A entrada para **sum** e **avg** deve ser numérica, mas os outros operadores podem operar em coleções de tipos de dados não numéricos, como strings.

<u>id_cliente</u>	nome_cliente	credito
1	José de Melo Bico	1000
2	Antônio Barbudinho	800
3	Waldick Soriano	1200
4	Tiririca da Silva	350

select **max(credito)** from cliente

Retorna 1200

select **min(credito)** from cliente

Retorna 350

select **sum(credito)** from cliente

Retorna 3350

Funções Agregadas

- A Cláusula GROUP BY é utilizada para agrupar linhas da tabela que compartilham os mesmos valores em todas as colunas da lista.

<u>id_saque</u>	data	nome_cliente	valor
1	10/11/2007	José de Melo Bico	300
2	10/11/2007	Antônio Barbudinho	120
3	11/11/2007	Waldick Soriano	50
4	11/11/2007	Tiririca da Silva	20
5	11/11/2007	Waldick Soriano	200
6	12/11/2007	José de Melo Bico	150

```
select nome_cliente, sum(valor) from saque group by  
nome_cliente
```

nome_cliente	sum
José de Melo Bico	450
Antônio Barbudinho	120
Waldick Soriano	250
Tiririca da Silva	20

Cláusula Having

- A cláusula HAVING restringe os resultados do GROUP BY.

```
select nome_cliente, sum(valor) from saque group by  
nome_cliente having sum(valor)>200
```

nome_cliente	sum
José de Melo Bico	450
Waldick Soriano	250

Teste de Valores Nulos

- **Relembrando:** valores nulos são aqueles em que não se aplica um valor para aquele atributo naquela entidade;
- Testamos se um valor é nulo ou não através da construção **is null** ou **is not null**, dependendo do caso.

```
select nome_cliente from cliente where numero_apto  
is null
```

Consultas Aninhadas

- A SQL permite testar registros que participam em outras consultas;
- O conectivo **in** testa a presença em uma consulta;
- O conectivo **not in** testa a ausência em uma consulta;

Consultas Aninhadas

- Exemplo:
 - cliente (id_cliente, nome_cliente);
 - dependente
(num_dependente, id_cliente, nome_dependente)
id_cliente referencia cliente
- Como listar todos os clientes que possuem dependentes?
- A consulta **select id_cliente from dependente** mostra o id de todos os cliente que possuem dependentes.

Consultas Aninhadas

select nome_cliente from cliente where **id_cliente** in
(select id_cliente from dependente)

<u>id_cliente</u>	nome_cliente
1	João
2	Cesimar
3	Gustavo
4	Thásio
5	Clodoaldo
6	Xuxu

<u>num_dependente</u>	<u>id_cliente</u>	nome_dependente
1	1	Rita Zero Hora
1	2	Toninha Caminhão
1	4	Tatá Jr.
1	3	Maria Gasolina
2	3	Guga Jr.

- A subconsulta **(select id_cliente from dependente)** retorna:

<u>id_cliente</u>
1
2
4
3
3

Consultas Aninhadas

```
select nome_cliente from cliente where id_cliente in  
(select id_cliente from dependente)
```

nome_cliente
João
Cesimar
Gustavo
Thásio

Junções

- Considere o esquema:
 - **Fornecedor**(id_fornecedor, nome_fantasia);
 - **Produto**(id_produto, id_fornecedor, descricao, preco)
id_fornecedor referencia Fornecedor
- Tabelas:

<u>id_fornecedor</u>	nome_fantasia
1	Seu Zé Distribuidora
2	Chico Tripa Atacado
3	Barbudinho LTDA
4	Xuxu Distribuidora

<u>id_produto</u>	id_fornecedor	descricao	valor
1	1	Bala	0.25
2	1	Chiclete	0.50
3	2	Pirulito	0.70
4	3	Picolé	1.00
5	3	Chocolate	1.20
6	3	Dadá	0.20

Junção Interna

```
select * from Fornecedor inner join Produto on  
Fornecedor.id_fornecedor=Produto.id_fornecedor
```

id_fornecedor	nome_fantasia	id_produto	id_fornecedor	Descricao	valor
1	Seu Zé Distribuidora	1	1	Bala	0.25
1	Seu Zé Distribuidora	2	1	Chiclete	0.50
2	Chico Tripa Atacado	3	2	Pirulito	0.70
3	Barbudinho LTDA	4	3	Picolé	1.00
3	Barbudinho LTDA	5	3	Chocolate	1.20
3	Barbudinho LTDA	6	3	Dadá	0.20

Junção Externa Esquerda

```
select * from Fornecedor left outer join Produto on  
Fornecedor.id_fornecedor=Produto.id_fornecedor
```

id_fornecedor	nome_fantasia	id_produto	id_fornecedor	Descricao	valor
1	Seu Zé Distribuidora	1	1	Bala	0.25
1	Seu Zé Distribuidora	2	1	Chiclete	0.50
2	Chico Tripa Atacado	3	2	Pirulito	0.70
3	Barbudinho LTDA	4	3	Picolé	1.00
3	Barbudinho LTDA	5	3	Chocolate	1.20
3	Barbudinho LTDA	6	3	Dadá	0.20
4	Xuxu Distribuidora	NULL	NULL	NULL	NULL

Os registros da tabelas à esquerda são inseridos, porém com os dados que não têm correspondência com a tabela da direita são preenchidos com NULL.

Outras Junções

- Junção Externa Direita (right outer join)
 - Completa com os registros não relacionados da tabelas à direita;

```
select * from Fornecedor right outer join Produto  
on Fornecedor.id_fornecedor=Produto.id_fornecedor
```

- Junção Externa Completa (full outer join)
 - Completa com os registros não relacionados da tabelas à direita e à esquerda;

Junção Natural

- Na junção natural (natural inner join), o resultado é semelhante ao inner join, porém sem atributos repetidos aparecendo na junção;

```
select * from Fornecedor natural inner join Produto
```

id_fornecedor	nome_fantasia	id_produto	Descricao	valor
1	Seu Zé Distribuidora	1	Bala	0.25
1	Seu Zé Distribuidora	2	Chiclete	0.50
2	Chico Tripa Atacado	3	Pirulito	0.70
3	Barbudinho LTDA	4	Picolé	1.00
3	Barbudinho LTDA	5	Chocolate	1.20
3	Barbudinho LTDA	6	Dadá	0.20

Exclusão de Registros

- Sintaxe:

```
delete from tabela where predicado
```

- A instrução acima deleta todos registros da tabela em que o predicado especificado seja verdadeiro;
- Exemplo:

```
delete from turma_professor where id_professor=4
```

Atualização de Registros

- Sintaxe:

```
update tabela set tabela.A1=novo_valor_A1,  
tabela.A2=novo_valor_A2,..., tabela.An=novo_valor_An where  
predicado
```

- A instrução acima atualiza todos os atributos A_1, \dots, A_n dos registros da tabela em que o predicado especificado seja verdadeiro;
- Exemplo:

```
update turma_professor set id_professor=5 where  
id_professor=4
```